Application Development

A collaboration application is a program that facilitates groups working together by collecting, organizing, distributing, and tracking information across an organization. When an application that was designed for personal use is placed in a public folder, it becomes available to others so that they can work together. An effective collaboration environment streamlines workflow so colleagues can interact efficiently, find and share information, collaborate on documents, and publish information to the company intranet or the Internet. This growing integration of Web-based technologies to build collaboration applications requires that you are more involved to ensure the reliability and integrity of mission critical operations. By understanding development terminology and basic programming procedures, you are in a much better position to help deploy applications while maintaining server policies and security.

This chapter explores the development capabilities of Exchange 2000 Server and the Microsoft Web Storage System. This Web Storage System provides easy access to its contents for developers' applications using Web protocols and versatile storage. Messages, files, Web content and semi-structured data can all be stored in the Web Storage System. The many protocols used by Exchange make it an ideal vehicle for developing applications. It is this versatility that makes Microsoft Exchange 2000 a rich development platform.

In this Chapter:

Overview of Web-based Collaboration

Exchange 2000 Web-based Collaboration Applications

Event Support in Exchange 2000

Workflow Components For Exchange 2000

Application Development Utilities

Overview of Web-based Collaboration

Microsoft Exchange 2000 Server provides the infrastructure for building collaboration applications. Exchange 2000 uses a variety of protocols, API components and formats to permit communication between applications and the Web Storage System.

Before reviewing how Exchange 2000 Server works at its most basic level, it is important to know what you can accomplish with Exchange 2000. For example, Exchange 2000 technology allows the construction of applications specific to your organization that can display functional in-boxes, display corporate information on Active Server Pages, while allowing users to track expense reports and other business data.

A more detailed example is an Exchange 2000 application called Microsoft Transit, which you can place at a Career Planning/Management Web Site. Transit demonstrates how you can use various features of the Web Storage System to create a career management tool. Transit can exist in the Human Resources space of a company and it

can use career maps created by users to navigate between personal and collaborative job spaces. In personal job spaces the user can create goals, respond to employee review requirements, collaborate with peers and direct reports, and explore career options within the company. In public job forum spaces, the user can interact with others with similar professional interests, engage in networking, and access professional development information.

The following are the protocols, components and formats used in Exchange 2000:

- Network Protocols
  - HTTP
  - World Wide Web Distributed, Authoring and Versioning (WebDav)
  - Simple Mail Transfer Protocol (SMTP)
  - Network News Transfer Protocol (NNTP)
  - Internet Message Access Protocol Version 4 (IMAP4)
  - Post Office Protocol version 3 (POP3)
  - Lightweight Directory Access Protocol (LDAP)
- API Components
  - ActiveX Data Objects (ADO)
  - Active Directory Services Interface (ADSI)
  - Collaboration Data Objects (CDO)
  - Object Linking and Embedding Data Bases (OleDB)
  - Messaging Application Programming Interface (MAPI)
- Formats
  - Request For Comments number 822 (RFC822))
  - Multipurpose Internet Mail Extensions (MIME)
  - Hypertext Markup Language (HTML)
  - Extensible Markup Language (XML)

Only processes running on the server can use the OLE DB provider. To access the Web Storage System contents remotely, you must use the protocols listed in the following section.

Network Protocols

Exchange 2000 network protocols are not located in the Web Storage System. They run as part of the IIS process. The advantage of removing the protocols from the Web Storage System is that it allows you to dedicate servers to specific tasks, such as managing the database or handling client requests. This removal of the protocols from the Web Storage System increases the flexibility, reliability and scalability of Exchange 2000.

HTTP and WebDAV Extension

Hyper Text Transfer Protocol (HTTP) is a way of connecting to the Internet. An HTTP client such as Microsoft Internet Explorer is a web browser that uses HTTP to access information on the Microsoft Internet Information Server (IIS). Outlook Web Access (OWA) enables Exchange 2000 to integrate with an IIS server, and connect to the Internet. This allows users to gain access to mail and public folders in the Web Storage System using a standard web browser. The three main HTTP functions used by the Web Storage System are GET, PUT and POST:

- **GET** retrieves a document
- **PUT** puts an item in a folder
- **POST** submits an item to a folder

The World Wide Web Distributed, Authoring and Versioning (WebDAV) protocol provides access to an extensible set of associated properties. WebDAV also defines protocol commands used to search, move, copy, delete, lock and unlock resources, and make new collections of resources (folders). The commands PropFind and PropPatch are used for property (or attribute) manipulation. These features make it an ideal protocol for creating interoperable, collaborative applications. Major features of the protocol are as follows:

- **Locking (concurrency control)**: Long-duration exclusive and shared write locks prevent the overwrite problem, where two or more collaborators write to the same resource without first merging changes. To achieve robust Internet-scale collaboration, where network connections might be disconnected arbitrarily, and for scalability, since each open connection consumes server resources, the duration of WebDAV locks is independent of any individual network connection.
- **Properties**: XML properties provide storage for metadata, such as a list of authors on Web resources. These properties can be set, deleted, and retrieved using the WebDAV protocol.
- **Namespace manipulation**: Since resources might need to be copied or moved as a Web site evolves, DAV supports copy and move operations. Collections, similar to file system directories, can be created and listed.

The main WebDAV functions used with the Web Storage System are as follows:

- **MKCOL** Makes a collection that in essence is the same as a Web Storage System folder. Therefore MKCOL allows you to make folders and set their properties.
- **LOCK** Locks documents to prevent writes.
- **UNLOCK** Allows writes and creates updates.
- **PROPFIND** Searches for properties either one at a time or for a whole set of folders.
- **SEARCH** The same features as PROPFIND, but in addition can add a filter, subquery and/or sort to the search.
- **PROPPATCH** Sets arbitrary properties on an item
- **SUBSCRIBE** A notification that you use to add a user to a list
- **UNSUBSCRIBE** A notification that you use to remove a user from a list

- **POLL** Checks to see if the notification is fired.

SMTP

Simple Mail Transfer Protocol (SMTP) is the Internet standard for transporting and delivering electronic messages. Based on specifications in Request for Comments (RFC) 821 and RFC 822, Microsoft SMTP Service is included in Windows 2000 and Exchange 2000 uses the primary SMTP address in the default recipient policy as your organization's default domain.

Microsoft Exchange 2000 Server expands Microsoft SMTP Service, enhancing the basic delivery functions of the protocol without affecting its compatibility with other messaging systems. Exchange gives you greater control over message routing and delivery, and provides secure access and channels for managing the service. Although you can make some configurations for sending e-mail to remote domains on the virtual server, you should do most of that administrative work at the SMTP Connector.

If you use Microsoft SMTP Service on a computer before installing Exchange 2000, any configurations you make to the operating system for SMTP are lost. Also, messages in the Pickup or Queue directories are not delivered. In Windows 2000, SMTP uses the subdirectories in the <root>\Inetpub\mailroot directory. While these folders aren't, SMTP uses the <root>\Exchsrvr\mailroot directory.

Multiple virtual servers can be helpful in certain situations. If you have different groups of users with varying security requirements or message-size needs. You might want to create additional virtual servers. Additional virtual servers are also helpful for managing different types of e-mail. For example, in a mail gateway, one dedicated virtual server can handle Internet e-mail, while another handles internal e-mail.

NNTP

Network News Transfer Protocol (NNTP) is an application protocol that is used over TCP/IP networks. NNTP defines a set of client and server commands used to access newsgroups.

Microsoft Exchange 2000 Server uses NNTP to enable Outlook 2000 users to participate in online discussions over the Internet. Exchange also enables users running client applications that support NNTP to access newsgroup public folders on computers running Exchange.

Users can read and post items, such as messages and documents, to NNTP newsgroups that are represented in Exchange as public folders; for example, scientists can exchange research information by posting messages to a newsgroup public folder for their area of interest. Other scientists around the world can read and respond to items in the newsgroup. Items in newsgroups can be replicated to Usenet host computers through news feeds.

Microsoft Exchange 2000 Server enables Outlook 2000 users to participate in online discussions over the Internet using Network News Transfer Protocol (NNTP) newsgroups. Exchange also enables users running client applications that support NNTP to access newsgroup public folders available on Exchange servers.

Users can read and post messages and documents to NNTP newsgroups, which are represented in Exchange as public folders; for example, you can exchange information

with your customers by posting frequently asked questions to a newsgroup public folder. Your internal sales support staff can answer questions and customers around the world can relate their own experiences. Items in the newsgroup can be replicated to Usenet host computers in news feeds.

IMAP4

Internet Message Access Protocol version 4 (IMAP4) is an Internet messaging protocol that enables a client to access mail on a server rather than downloading it to the user's computer. IMAP4 is designed for an environment in which a user logs on to the server from several different workstations. In such an environment, downloading a user's mail to a specific computer is usually impractical because the user does not always use the same computer. For example, IMAP4 is widely used in universities where students connect to the mail server from different labs throughout the campus. Once connected, users can access their mailboxes as though their mail is stored locally. IMAP4 does not handle mail transport. Simple Mail Transfer Protocol (SMTP) handles this feature.

IMAP4 allows a client to access private messages and public folders on a server. Users with an IMAP4 client can access mail in their Microsoft Exchange 2000 Server mailboxes without downloading the entire mailbox to a specific computer.

A single client can access multiple mailboxes to retrieve specific messages or portions of a message, such as an attachment. IMAP4 clients can also search a mailbox and store flags to identify messages that have been read.

You can configure your IMAP4 server to grant or deny access to specific computers, groups of computers, or domains. You can grant or deny access to a single computer based on an IP address or by overriding IMAP access on a per-user basis. A group of computers can either receive or be denied access based upon their subnet address and subnet mask. You can also control access to an entire domain at once by specifying a domain name.

POP3

Post Office Protocol version 3 (POP3) is an Internet protocol that allows a POP3 client to download mail from a server. This protocol works well for computers that cannot maintain a continuous connection to a server.

Both IMAP and POP3 are Internet messaging protocols that allow clients to access their mail. The difference between these protocols is where the client manipulates their messages. IMAP allows a client to access and manage mail on a server. POP3 allows a client to download mail from an Inbox on a server to the client computer where messages are managed.

Unlike Internet Message Access Protocol version 4 (IMAP4), POP3 does not allow users to manipulate messages on the server. Mail simply downloads to the client where messages are managed. POP3 provides access only to a user's Inbox; it does not support access to public folders.

All management of POP3 user mailboxes is performed within Active Directory. In previous versions of Exchange, Exchange uses its own database to hold directory information, and recipients are managed within the Exchange Administrator. The directory database provided in Windows 2000 is capable of holding all messaging directory data. Because a separate directory for Exchange is not necessary, all

manipulation of mailboxes is coordinated within the Directory Service Manager. After installing System Manager on the Windows 2000 computer, a set of extensions is added to the standard Active Directory console. This allows an Exchange mailbox to be created when a new user account is defined in Active Directory Users and Computers.

Only users with Windows 2000 accounts can have a mailbox and to send and receive mail. If an account is mail-enabled but not mailbox-enabled, users can only send mail. Mailbox-enabled users can also receive mail and configure additional settings.

A unified namespace provides easier administration of multiple POP3 servers. For example, if you have three separate computers running a POP3 virtual server, you normally divide the user load by configuring certain users to connect to POP3Server1, other users to connect to POP3Server2, and others to connect to POP3Server3. If all POP3 servers are part of a front-end/back-end configuration, there is one name that provides user access to all POP3 servers in your configuration. You can configure clients to connect using the same POP3Server. You can use software or hardware load balancing to randomly distribute the load to any of the three POP3 servers. When you want to move a user's mailbox from one server to another, the client does not need to reconfigure the name of the server it logs on to. As your user population grows, you can add another computer to the front-end bank of servers without reconfiguring the client.

When connections are made using the Secure Sockets Layer (SSL), information is encrypted and decrypted. This encryption and decryption is processor intensive and can affect performance. If your POP3 virtual servers are deployed in a front-end/back-end configuration, the front-end server can process encryption with the client. When the front-end server and back-end server communicate, they do so without the Network bandwidth load of SSL encryption. This reduces the load on the back-end server.

LDAP

The LDAP protocol is a distributed, hierarchical directory service protocol that you use to gain access to repositories of users and other network objects. Because LDAP typically is not tightly integrated with the host operating system, information can be kept both in LDAP and in a name service such as the Network Information Service. When you use LDAP to gain access to the groups, you reduce redundancy and maximize LDAP's scalability.

LDAP gives you the option of using paged or non-paged results when performing a search. If you use non-paged, then the maximum number of objects returned is limited by the MaxPageSize constraint on the domain server. If you use paged results, then you can return all the results. You set the LDAP client page size in the Search options dialog. This value cannot exceed the MaxPageSize constraint imposed by the domain server.

If you use the view tree option, then LDAP does not use paged results, and is limited to displaying the objects limited by the MaxPageSize constraint in the scope pane.

> **Note** The domain server imposes a limit on the maximum number of objects returned by a query in a single page. This is determined by the value of the MaxPageSize constraint in the LDAP Query Policy. The default value is 1000 and can be altered by changing the value in the default query policy or creating & assigning a new policy to the domain control. This limit means that any LDAP client can only retrieve a maximum of 1000 objects if it does not use paged results, or a maximum of 1000 objects per page (multiple pages obviously) if it uses paged results.

For more information about Network Protocols see "Exchange 2000 Architecture" in this book.

API Components

Many of the features of Exchange 2000 are made possible by the Windows Application Programmer's Interface (API) Components. These components are API function libraries that are composed of Dynamic Link Libraries (DLLs). These DLLs contain the actual function calls that the developer can use through code. These functions can enhance the application while allowing the developer to reuse programming code and save time.

The API components used in Exchange are:

- ADO
- ADSI
- CDO
- OLE DB
- MAPI

ADO

The developer uses Microsoft ActiveX Data Objects (ADO) to navigate through the Web Storage System. ADO allows the developer to access all the Microsoft Database APIs, as well as search for, and bind to objects. ADO contains built-in objects that the developer can use to add features to the application. This permits developers to add dynamic objects to a static web page.

Fig. X.X below shows the ADO Object Collection:

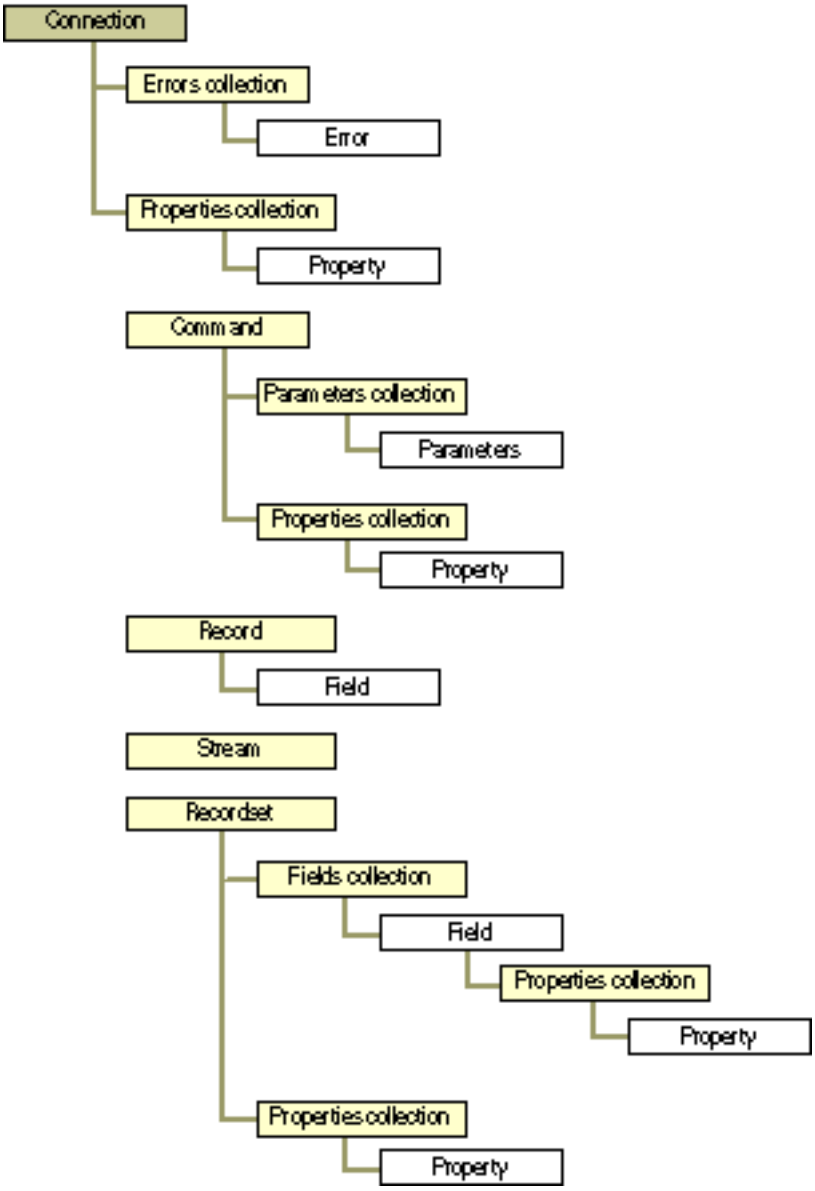

Figure X.X. The ADO Object Collection.

At the top of the ADO Object Collection diagram is the Connection object. It establishes and maintains a connection to a database or other data source. The Open function of this object establishes the actual connection. This object manages transactions in applications. Beneath the Connection Object is the Error Object. It contains information on an error generated during an operation or command. A Connection Object maintains a collection of Error Objects.

Descending down the diagram, the next object is the Command object. This provides a command in binary form.

Next is the Parameter object. This object enables the code to set or get a parameter that is used with a Command object. The Command object maintains a collection of Parameter objects.

Following is the Record Object. This object opens an item or folder in the Web Storage System using the item's URL. This object maintains a collection of fields.

The next object is the Stream Object. It accesses resources as bits of streaming data, so that when you open e-mail attachments, instead of opening a large portion of data at once, the data is streamed. This results in better performance.

The next object on the diagram is the Recordset object. This object views one record at a time and provides functions for traversing a set of records. This object's Open function provides access to a record set by an SQL statement, table name or a command object and a connection object.

Following is the Field Object. It provides access to a Recordset object's field. Modifying the field value modifies the actual data in the data source. A Recordset object maintains a collection of Field objects.

At the bottom of the diagram is the Property object. It provides access to a property of the Field, Command, Connection, and Recordset objects. All of these objects maintain collections of Property Objects. Property objects support dynamic properties, enabling a data source to add properties to an object at run time.

Of all these objects, the Connection object is the most important. If an application slows down the network, it may be that the developer did not use the same ADO Connection for multiple operations. Creating new connections throughout an application is both expensive and inefficient, especially when deploying Active Server Page (ASP) applications. The developer should make the connection object at the beginning of the session. That object is then referred to throughout all of the subsequent ADO commands. By working with developers, you can bring up and resolve network resource issues in the early stages of application design.

Querying the Web Storage System With ADO

To retrieve specific information from the Web Storage System, you can use ADO to query only those records that you need to work with. You need to establish a connection explicitly with the Web Storage System.

You begin by building the core SQL Select statement with the Web Storage System URL of the folder you want to query. Table X.X describes the SQL commands that are supported and unsupported in the Web Storage System:

Supported SQL Commands	Unsupported SQL Commands
SELECT, SELECT *	JOIN
LIKE	MAX
WHERE	MIN
ORDER BY	SUM
GROUP BY	AVG
CONTAINS	
FREETEXT	
Column Aliasing	

Table x.x; Supported and Unsupported SQL Commands in the Web Storage System.

> **Note** The commands CONTAINS and FREETEXT only work if content indexing is turned on and what you search for is indexed

You need to specify the scope of the query in the **From** clause. **SCOPE** tells the Web Storage System how deep you want to look for information. This can be either **DEEP** (this folder and all subfolders) or **SHALLOW** (this folder only) from the URL you specify. **SHALLOW** is the default.

The MAPI Hierarchy (also called a MAPI tree) only supports **SHALLOW** traversals when. You can perform deep traversals on non-MAPI Public folders that you create. The following is an example of an ADO query against the Web Storage System:

The **SELECT** statement takes two properties, the subject of a mail message and a custom property called financial:

```
SELECT “urn:schemas:httpmail:subject”, “urn:schemas:mydomain:financial”
```

The **FROM** clause requires the keywords Shallow or Deep followed by Traversal Of and the URL of the folder.

```
FROM SCOPE ( ‘shallow traversal of file://backofficestorage/./public folders/accounts’)
```

Rather than bring back everything that matches, you can use the **WHERE** clause to restrict the returned records to those matching the subject

```
WHERE “urn:schemas:httpmail:subject” = ‘loan’
```

When the query executes, the results are returned in a recordset object. You can then perform whatever action you need to, such as copy or navigate through the contents.

Using ADO to Access Objects Through the ADSI Provider

The Active Directory Service Interface’s (ADSI) provider implements OLEDB interfaces that allow you to use ActiveX Data Objects (ADO) to access objects in compliant directories. The ADSI provider is read only when used with ADO. You must create an ADO connection object and set its Provider property to **"ADsDSOObject"**. You can specify any string, including **""**, as the first argument connection string of the ADO connection object's open function.

The connection object Execute function's CommandText (first object) is an LDAP query composed of four elements separated by semicolons, in the following format:

```
<LDAP://server/adsidn>;ldapfilter;attributescsv;scope
```

Where:

- **server** is the name (or IP address) of the server hosting the directory.
- **adsidn** is the distinguished name of the starting point for your query expressed in ADsPath format with "/" separators and the root of the namespace to the left. You can also use an X-500 style attributed name format with the relative distinguished names separated by commas and the root of the name space to the right.
- **ldapfilter** is the LDAP filter string.

- **attributescsv** is a comma-separated list of names of the attributes to be returned for each row in the recordset.
- **scope** is either: base, one level, or subtree.

To return the ADsPath, class, and cn attributes of all objects in the recipient containers in an Exchange server, you can use the following text (in URL format):

```
LDAP:<//server/o=organization/ou=site/cn=recipients>;
   (objectClass=*);ADsPath,objectClass,cn;subtree"
```

or, in attributed name format:

```
<LDAP://server/cn=recipients,ou=site,o=organization>, _
   (objectClass=*);ADsPath,objectClass;subtree
```

The following Visual Basic sample code illustrates this query:

```
Dim conn As ADODB.Connection
    Dim rs As ADODB.Recordset
    Set conn = New ADODB.Connection
    conn.Provider = "ADSDSOObject"
    conn.Open "ADs Provider"
    Set rs = conn.Execute( _
          "<LDAP://server/o=organization/ou=site/cn=recipients>; _
          (objectClass=*);ADsPath,objectClass,cn;subtree")
    While Not rs.EOF
       Debug.Print rs.Fields(0).Value, rs.Fields(1).Value, _
             rs.Fields(2).Value
       rs.MoveNext
    Wend
    conn.Close
```

ADSI

You use ADSI is used to add new users, modify attributes and browse Active Directory, as well as other activities. You can use ADSI to govern the lifecycle of abstract directory objects (such as, computers and user groups) as well as access these objects even if they are located on a different operating system. Using ADSI, you can update user information from a web page and distribute it across a multi-operating system network.

ADSI uses the capabilities of directory services from different network providers to present a single set of directory service interfaces for accessing and managing network resources. You can use ADSI services to number and manage resources in a directory service, regardless of which network environment contains the resource. This can be an LDAP-based, NDS-based, or NTDS-based directory.

ADSI is a standard Windows-based interface for meta-directory applications. A meta-directory is a high-level network directory service designed to unite account and

resource information from multiple network operating environments. The goal of ADSI is to synchronize this information across differing directories. Meta-directories work by extracting security and account data from each directory and then managing this information as part of an external database. ADSI is comprised of a series of client-side DLLs that provide a common set of directory management functions. You can access these functions from almost any environment. As a result, you can add or remove user accounts, configure shared resources and browse the directory tree; all from a single, integrated console. Exchange 2000 uses ADSI to integrate application services with Active Directory's security and account functions. You can also access ADSI using VBScript and Windows Script Host, to perform directory maintenance tasks.

ADSI allows application developers to query and update Active Directory. ADSI provides an abstraction of directory service contents, which makes it independent of the underlying directory service. The APIs and interfaces in ADSI conform to COM and support standard COM features including automation.

ADSI allows different applications to communicate locally, which enables developers to create directory-enabled applications. ADSI is built on a provider-based model. While clients use the COM interfaces exposed by ADSI, providers implement the mapping between those COM interfaces and the underlying directory system. In this provider-based model, it can serve as the interface to a number of directory services, accessing each one through its own provider. This eliminates concerns about the underlying differences between various directory implementations or namespaces.

Because ADSI is a set of COM objects, you can use it with Microsoft Visual Basic or JScript to make Web-based applications directory-enabled. ADSI creates a directory structure in which you access any directory service using the same universal commands. This ADSI directory structure contains the following two basic types of objects:

- **Container** This object holds a number of other objects as either children or members.
- **Leaf** This object has neither children nor members.

The foundation object for this structure is the container named **ADS://.** This container stores a number of namespace containers. Each namespace represents a particular syntax that you use to gain access to a target directory service.

> **Note** The target service is not found in a namespace unless it has an ADSI provider installed or you write one of your own and give it a namespace.

Retrieving Information With ADSI

When you look for information, you must know if the information is contained in the object instance or in the object’s schema definition. The object name is contained within the object instance but a list of properties about the object is contained within the object’s schema. The following example shows both an object instance and the object’s schema definition:

```
Used Car Instance

Name: Horseless Carriage

Year Built: 1899
```

```
Engine: Steam

Used Car Object Instance

Attribute "Horseless Carriage", single value, data-type "text"

Attribute "Year Built", single value, data-type "Integer"

Attribute "Engine", single value, data-type "text"
```

In this example the car's name, year built, and engine information are kept in the instance of a used car object. The schema contains definitions of the used car attributes such as data-type and engine-type constraints

In comparison, with SQL the object definition (or object schema) is similar to a table definition and the object instance is similar to a particular row in the table. If the information crosses more than one table, then in SQL the developer is responsible for managing those relationships. Using ADSI, the interface is responsible for the same relationships.

In ADSI each object, regardless if it is an instance object or a schema object, has these properties: Name, Parent, Schema, Class, GUID, and ADSPath. These are all the objects you need to know in order to navigate through the hierarchical structure of the database. The Name property is the name of the object. The Parent property gives the location of the parent object. The Schema property governs the object. The Class property is the type of object. The class contains all the attributes that an object can have. The GUID property is the unique identifier for the object. The ADSPath is the location of the object you retrieve.

To get the property information for the default database, you must either know the name of the property, or get the list of properties for the class. In the car example shown earlier, you have all the properties and their values. You can specify any object to have mandatory or optional properties. You must set all mandatory properties to create an object.

Using ADSI you can create a temporary object outside of the source and set properties without affecting the performance of other requests to the data source. After you set the properties, you add the object to the database source with the object.SetInfo statement. This is also true for the Update command. Using ADSI, you retrieve the object once, set the properties, and then make comments about the updates with an object.SetInfo statement.

If you want to change or monitor a particular item in the directory structure, you need to bind an object to that item.

For example, if you want to change a user account, use the following syntax:

```
Set objUser = GetObject("WinNT://microsoft.com/Bob")
```

In this example, Microsoft.com is the name of the local domain and Bob is the name of the account.

You can access the six core properties with the object/function syntax, and print them on your screen. When you are bound to a User object, you also have access to the properties and functions defined by the ADSI User object. The number of properties varies from one namespace provider to another.

It is important to know how to enumerate the objects within a container. In this case, the code varies slightly depending on whether the container objects are members or children of the container. For example, groups are children of a domain and users are members of a group.

The code to enumerate the user Executives of the GlobalHQ group is as follows:

```
<HTML>
<BODY>
<%
  Dim oGroup
  Dim oExecutive
  Set oGroup = GetObject("WinNT://microsoft.com/GlobalHQ")
  For Each oExecutive in oGroup.Executive
    Response.Write oExecutive.Name & "<BR>"
  Next
%>
</BODY>
</HTML>
```

To list the children of an object, use a "For Each...Next" loop:

```
Set oContainer = GetObject("WinNT://Microsoft.com")
  For Each oChild in oContainer
    Response.Write oChild.Name & "<BR>"
  Next
```

These are the two important coding functions for any ADSI page. You can also write an ASP to number the members of the Mandatory Properties and Optional Properties of the schema for that object's class.

Note ADSI can sometimes generate an unexpected error message when sending a message. When there are no mandatory properties to number, you should use the On Error Resume Next property to avoid generating an error message and stopping the program.

For more information about what properties and functions each object class supports, see ADSI Help.

CDO

To manipulate an item in the Web Storage System, you use the Collaboration Data Object (CDO) library to encapsulate difficult or repetitive operations, such as meeting requests. CDO is an API specification that defines objects, interfaces, functions, and properties for accessing messaging and collaboration data; and is also available in a series of object libraries. CDO for Windows 2000 is an SMTP-based library that provides base-messaging features. CDO comes with Windows 2000 Workstation and Server. You can use these CDO objects in applications, and save developers the time-consuming

task of re-inventing commonly used components. You can use CDO for Windows 2000 to send e-mail messages and post discussion and news messages where no mailbox is required. It also provides extensive MIME content creation and management and support for the SMTP and NNTP protocol stacks. You can use Windows 2000 CDO for mail-intensive applications and any application needing to construct MIME content. CDO itself is not remotely accessible.

When Exchange 2000 installs, it adds an extensive set of CDO objects for Exchange object types and provides a larger set of objects than Windows 2000 CDO provides. Exchange 2000 CDO is compatible with all existing applications, including CDO 1.2x, MAPI, and Windows NT Server CDO. Exchange 2000 CDO provides objects for scheduling systems, resource booking, as well as support for e-mail, mailbox, voice-mail, fax, and pagers. You can also use Exchange 2000 CDO for server-based agents. CDO is an integral part of Microsoft’s Rapid Application Development (RAD) strategy.

Exchange 2000 CDO works with ADO and OLEDB. You can use ADO and OLE DB for all generalized data access, such as rowsets, queries, hierarchy navigation, and ACLs. You can use recordsets and rowsets in place of certain collections, such as Items, Folders, Recipients, Attachments, and Attendees.

You use CDO to access collaboration-specific data, such as Exchange-specific objects. CDO adopts Internet standards for content and protocols at the API level, including the following:

- RFC822
- RFC977
- RFC1036
- MIME
- S/MIME
- HTML
- MHTML
- ICalendar (internet standard for exchanging calendaring information)
- vCard
- SMTP
- NNTP
- LDAP
- HTTP

The purpose of CDO is to provide a consistent programming model with the same objects and interfaces on the client and server side. For example, you can use the Outlook object model on the client side and CDO on the server side, and use the same application logic, behavior, and features across all applications.

Exchange 2000 CDO provides a convenient object model for managing folders, messages, Exchange mailboxes, appointments, contacts, and other items as well as the properties of those items. CDO is designed to operate in the Web Storage System and

supports Internet standards and protocols. It provides direct access to OLE DB interfaces for using the Web Storage System.

CDO integrates with the Microsoft ActiveX Data Objects (ADO) 2.5 component to provide you with a consistent data-access interface to the Web Storage System and the Windows 2000 Active Directory. As you set values to CDO properties or access Web Storage System properties, CDO saves data to the correct Web Storage System or Active Directory locations.

Using IDataSource

CDO objects that have data have an IDataSource interface. This interface manages the data of a resource. The data is accessed as a property.

To open an existing item use the following syntax:

```
prsExist.IDataSource.Open strURL, , _
   adModeReadWrite
```

To save a new item use the following syntax:

```
prsNew.IDataSource.SaveTo strURL
```

Creating Contact Information

The CDO Person class represents contact information. CDO properties exist for most contact fields. You use the DataSource SaveTo or SaveToContainer function to save the contact. For example:

```
strURLFldFilms = conURLDomain & _
 "/public folders/Movie Production/microsoft/"
Set prsNew = New CDO.Person
With prsNew
    .FirstName = "Kim"
    .LastName = "Yoshida"
    .JobTitle = "Director"
    .HomeCity = "Los Angeles"
    .HomeState = "California"
    .Email = "kimy@microsoft.com"
    .DataSource.SaveToContainer
   strURLFldDoctors, , , adCreateOverwrite
End With
```

Scheduling

Scheduling uses Appointment, CalendarMessage, Attendee, DataSource, and Configuration. To perform scheduling in CDO, you must begin by configuring for the organizer. Then you set appointment properties, invite the attendees, save to a folder container and send a meeting request. For Example:

```
Set appt = New CDO.Appointment
```

```
With appt
    .Configuration.Fields(cdoSendEmailAddress) = _
     "someone@microsoft.com"
    .Configuration.Fields.Update
    .StartTime = #10/9/1999 12:30:00 PM#
    .EndTime = #10/9/1999 1:30:00 PM#
    .Subject = "Exchange 2000 Web Storage System – The Musical"
    .Location = "Film Theatre“
    Set att = .Attendees.Add
    att.Address = "bob@microsoft.com"
    att.Role = cdoRequiredParticipant
        Set msg = .CreateRequest
    msg.Message.Send
        .DataSource.SaveToContainer strURLContainer
End With
```

Sending Mail

To send mail you must have a sender and a recipient. Begin by configuring for the sender and then address with To, CC, or BCC properties. Separate multiple addresses with commas and set the subject and TextBody properties for the test message. Use the Send function to send it. For example:

```
Set msg = New CDO.Message
With msg
   .Configuration _
      .Fields(cdoSendEmailAddress) = "someone@microsoft.com"
    .Configuration.Fields.Update
    .To = "someone@microsoft.com"
    .Subject = "Movie Awards"
    .TextBody = "We’ve won an Oscar!"
    .Send
End With
```

Deciding When to Use ADO and CDO

You use ADO to navigate and manipulate records, set basic properties and save generic items. You use Windows 2000 CDO for mathematical problems and repetitive operations.

OLEDB

The OLEDB API component provides a scriptable interface. Formerly known as Object Linking and Embedding Data Bases, OLE DB is the Microsoft technology that permits the sharing of multimedia data among servers, while allowing applications and Microsoft software on those servers to exchange information and work together. Exchange 2000's OLE DB provider allows access to the Web Storage System using ADO.

MAPI

Extended MAPI (Messaging Application Programming Interface) version 1.0 is an industry-wide standard for writing messages, e-mail, and workflow applications. Exchange 2000 and many other common messaging programs use MAPI. You use MAPI interfaces to create and access diverse messaging applications and messaging systems, offering a uniform environment for development and use and providing true independence for both.

If you can call an entry point in a DLL, and pass a data record conforming to an externally defined structure, then you can use simple MAPI. If you can implement or manipulate arbitrary objects conforming to the Microsoft Component Object Model (COM), then you can use full MAPI.

Formats

Formats specify the properties, particularly visible properties, of an object. For example, word processing applications allow you to format text, which involves specifying the font, alignment, margins, and other properties. Exchange 2000 uses the following formats :

- RFC 822
- MIME
- HTML
- XML

RFC 822

Request For Comments (RFC) 822 is a message representation protocol which specifies details about message headers, but which leaves the message content, or message body, as ASCII text. In the TCP/IP protocol suite, electronic mail is in the RFC 822 format.

While SMTP has remained unchanged over the years, the Internet community has made several changes in the way it uses SMTP. In particular, the conversion to the Domain Name System (DNS) has caused changes in address formats and in mail routing. RFC-822 specifies the Internet standard format for electronic mail messages. RFC-822 replaces RFC-733, which is an older standard that some organizations might use. The two formats are sometimes referred to by their numbers 822 or 733.

Some non-Internet mail environments with mail transfer protocols other than SMTP use RFC-822. SMTP is adapted for use in some non-Internet environments.

MIME

Multipurpose Internet Mail Extensions (MIME) is the standard protocol for including non-text information in e-mail, such as binary data, audio data, video data, and foreign language text that cannot be represented in ASCII text. MIME supports the transmission of mixed-media messages across TCP/IP networks.

HTML

HyperText Markup Language (HTML) is the authoring language used to create documents on the World Wide Web. HTML defines the structure and layout of a Web document by using a variety of tags and attributes.

XML

You use Extensible Markup Language (XML), to receive and respond to Internet data requests. XML is a foundation for building applications. You can easily create XML files in the Web Storage System. The XML Data Object Model can save to a Web Storage System URL using the object's Save function. You can create the Xml Data Object Model objects based on Web Storage System properties and end-user choices from a simple HTML form. You can also store XML as the value for a Web Storage System property.

Extending Exchange data with custom data is simple. You and can combine it with E-mail, calendaring (a protocol used with CDO), tasks and contacts. You can use XML to add important metadata to documents, which increases their ability to be re-used and more easily managed. You can also use to define data structurally in a platform and language- independent manner.

By convention, property names for Web Storage System items are prefixed with a Uniform Resource Identifier (URI). These URIs are a set of unique names used to ensure that the Web Storage System renders globally unique names. Item properties are requested and returned in WebDAV sessions as XML elements that use URI namespaces. A URI for declaring namespaces can either be a URL or a URN.

For searches from Web Storage System clients, the SQL query can be sent in an Internet request using the XMLHttpRequest object. The XML parser imports the document into a dynamic data structure that can be recursively traversed. The HTTP 1.1 Search function sends the SQL query as the value of an XML node object. A returned The XML Data Object Model object with the search results can then be manipulated.

Microsoft Web Storage System

The Web Storage System is a semi-structured, hierarchical database that stores and manages documents, e-mail messages, Web pages, multimedia streams, data elements, and other items. Developers can access the Web Storage System using any programming language that supports COM and all the Exchange protocols listed earlier in this chapter.

Similar to file system directories and files, the Web Storage System is a hierarchy of folders. Each folder in the Web Storage System is a collection of items, including other folders. These folders and items are accessible by URLs.

With the Web Storage System, you can store an item and maintain property information about that item, including controlling its security and display. These properties are

actual fields on the item's row in the Web Storage System. An item's body (if present), whether text or binary, is itself a field on the row and is accessible as a stream. You can use ActiveX Data Objects (ADO) to navigate the Web Storage System hierarchy and obtain data using Record and Recordset objects. You use ADO or OLE DB to navigate and obtain objects from Web Storage System items, as well as create, delete, move, and copy folders and items. You can also manipulate data using HTTPDAV. This allows the user to add, copy, modify, move and search for items.

Web Storage System Schema

The Web Storage System defines a schema for the formats described earlier in this chapter. You can define a custom set of properties for your items and control their use. The content class property indicates the intent or purpose of the item. Therefore a property does not need to have a previously defined schema to be added to the Web Storage System. The Web Storage System schema provides the means for Web applications and search queries to efficiently access your data.

IIS Integration

Exchange 2000 fully integrates with the Internet because of the protocol support in IIS. Between IIS and Exchange there is a specialized high-speed link. When a request is received from the network by IIS, it determines which virtual path is requested. For example, if there is a request against http://server/exchange, the Exchange virtual path route's ISAPI register (DAVX.dll) responds to the request. That dll in turn can communicate over a dedicated high-speed channel to an http address that resides in the Web Storage System process. The http address takes the request and looks into the Web Storage System. Depending on the nature of the request, the address returns a row set, delete, or move to an item, and then returns a response to the Exchange ISAPI register. The register then takes the results of the request, if necessary renders the data, and then returns that back to IIS, or it returns a response code to the user.

It is possible to direct IIS to gain access to the Web Storage System using the WIN32 file system, but this only allows the user to add or delete files. Fig X.X illustrates the network request process using a loan approval ASP page as an example.

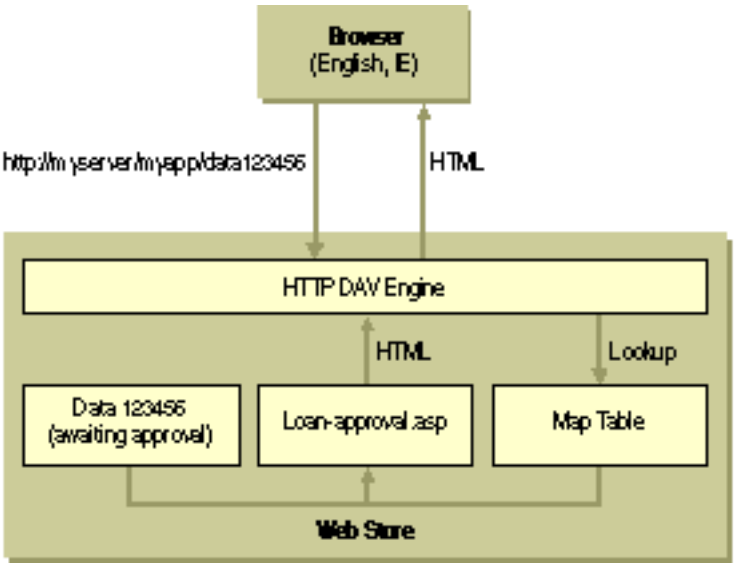

Fig. X.X Network Request Process Flow of A Loan Approval ASP Page

The Exchange Server Active Directory Schema

Active Directory provides you with a way of extending the schema unlike the Exchange 5.5 Server Schema, which does not allow you to add new classes or attributes. To write applications that access Active Directory, it is useful to have an understanding of the Active Directory schema. The directory schema is a collection of descriptions, or rules, that define characteristics of objects in the directory. The schema defines available object classes in the directory, relationships between object classes, attributes of each object class, and specific characteristics of attributes and classes. For more information about Active Directory Services, go to http://www.Microsoft.com/Exchange.

Exchange 2000 Web-based Collaboration Applications

Exchange 2000 enables developers to create web-based collaboration applications. These applications allow users to make full use of Exchange 2000 by using Internet technology.

The Microsoft Web-based Collaboration Applications are as follows:

- Microsoft Windows 2000 Server
- Active Server Pages
- Microsoft Internet Explorer
- Microsoft Internet Information Services
- Microsoft Outlook 2000

Unified through Windows 2000 Active Directory, these technologies give developers the ability to rapidly develop web-based applications.

Microsoft Windows 2000 Server

Exchange 2000 is integrated with Windows 2000 Server to take advantage of its security features, stability, and scalability. Windows 2000 Server and Workstation comes with CDO. Microsoft Exchange 2000 server comes with it’s own CDO library that provides a superset for the CDO library in Windows 2000.

Microsoft Internet Information Services

Microsoft Internet Information Services (IIS) simplifies communications and delivering Web sites to web browsers. IIS is the only Web server integrated into Windows 2000 Server. IIS is the front-end for Internet protocols in Exchange 2000. Protocol plug-ins in IIS communicate with Exchange 2000 Server about incoming and outgoing mail. This integration is possible by using the Exchange 2000 Installable File System (ExIFS), which allows large amounts of data to quickly pass between IIS and Exchange. It supports the Internet Server API (ISAPI). ISAPI enables programmers to develop Web-based applications that run much faster than conventional CGI programs because they are more tightly integrated with the Web server. ISAPI extends Internet servers, such as IIS,

using extensions and filters. Using an ISAPI extension, a browser typically sends information entered in a form, and the extension returns a complete HTML page built from code. In an ISAPI filter, all browser data, both inbound and outbound, can be modified before or after the server processes it. Extension and filter DLLs must conform to a specification. For the DLL to work there are specific functions that must be implemented.

In previous versions of Exchange, the information store manages the databases and client access protocols such as IMAP4, POP3, MAPI, and NNTP. In Exchange 2000, the Internet access protocols are removed from the store and are managed by IIS instead. Deploying a front-end/back-end configuration makes it possible to manage the Internet access protocols on a server that is separate from the one on which the store and databases run. Essentially, a bank of protocol servers handles the incoming client connections while the store servers are dedicated to running the databases. The main benefits of a front-end/back-end configuration are a unified namespace and reduced network bandwidth traffic for Secure Sockets Layer (SSL) encryption.

Active Server Pages (ASP)

An Active Server Page (ASP) is an HTML page with server-executed script embedded in the page. ASP is an open architecture application that does not need to be compiled. This makes it possible for users to combine HTML, scripts, and reusable ActiveX server components to create Web-based collaboration applications. ASP enables server scripting for IIS with native support for both Microsoft Visual Basic Scripting Edition (VBScript) and Microsoft Jscript.

When IIS receives a request for a page with an .asp extension, script within the HTML page is interpreted and presented back to the browser as HTML. Script is designated by <SCRIPT> and <%..%> tags. Any Active Scripting language can be used as the script language, but the most common the languages are either VBScript or Jscript. Scripts are not stored in a compiled form. They are interpreted when the .asp file is requested from the server.

An asp page can run under the context of the user requesting the page, the IIS_USER account or another account if configured. A common scenario is to require Windows 2000 authentication for the page so it runs when the user makes the request. In this model the user running the page must have the access control permissions to do the operation. The following is a sample ASP page using ADO to search the Web Storage System:

```
<%@ Language=VBScript %>
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
</HEAD>
<BODY>
<%
Dim rec 'As ADODB.Record
Dim rst 'As ADODB.Recordset
```

```
Dim urlFolder 'As String

‘Build the URL
urlFolder = _
"file://./backofficestorage/nwtraders/microsoft.com/Management/"

'Create a Record object
Set rec = server.createobject("ADODB.Record")

'Open the Management application folder
rec.Open urlFolder

'Extract the contents of the folder
Set rst = rec.GetChildren
%>

<h1>Folder List:</h1>
<hr>

<%
' Read the URL for each child record:
Do Until rst.EOF %>
   <%=rst.Fields("DAV:href")%><p>
   <%rst.MoveNext
Loop

'Close the objects and release memory
rst.Close
rec.Close
Set rst = Nothing
Set rec = Nothing
%>

</BODY>
</HTML>
```

Microsoft Internet Explorer

Internet Explorer 5.0 is a key component of Web-based application development in Exchange 2000. You can access the Web Storage System in Exchange using a Web browser such as Internet Explorer. By providing a URL, you can browse from the Internet straight to an item. The following are examples of items to which you can browse:

Private mailbox folder:

`http://domainname/exchange/yourname/inbox`

Public store:

`http://domainname/public/someFolder`

Mailbox item:

`http://domainname/exchange/yourname/inbox/hello.eml`

Always type the domain name in a front end – back end server configuration.

The Web Storage System renders an HTML 3.2 view of a folder's items. You can enhance these views into a user interface using Web Storage System Forms.

Microsoft Internet Explorer version 5.0 includes the XML COM component, a component containing an XML parser and related tools. Internet Explorer 5 has an object model that works with XML to communicate with httpDAV (also known as WebDAV) servers.

SMTP and NNTP Inbound and Outbound Events

SMTP and NNTP Inbound and Outbound Events refer to information that is destined for or leaving Exchange, respectively. With Inbound Events IIS creates a new file in ExIFS and writes to it. ExIFS returns a list of pages to which the message was written. The list of pages is passed to the Exchange Server process where the Extensible Server Engine (ESE) commits the pages by logging the information and page checksums (used to check data integrity) are stored in the EDB file.

In Outbound Events the Exchange Server gets the list of pages referenced by a message and the list of pages is passed to IIS. In turn, IIS opens a file in ExIFS using the list of pages. The message can now be transmitted quickly.

Note Page checksums are not verified during transmission.

Space Allocation

Space allocation refers to making space available in the database. The ESE knows which pages in the streaming file are currently committed. These pages are reserved and given to ExIFS. It allocates space for new messages from its reserved space and requests space from ESE when necessary.

File Semantics

File semantics refers to the wording used in calling files. ExIFS can reflect WIN32 file calls into the store. A call such as FindFirstFile can look in a mailbox folder for a list of messages. ExIFS can expose the contents of a database as ordinary files

Note The metabase stores configuration information in IIS version 4 and IIS version 5. You gain access to the metabase by using ADSI (including Script). All SMTP configurations are stored in the metabase.

Microsoft Outlook 2000

Outlook 2000 provides tools for creating programming and nonprogramming collaboration applications. The Desktop Information Manager included in Outlook 2000 centralizes all communication, organization, and management of information. Outlook 2000 supports multiple e-mail providers, scheduling, task management, journals, contacts, and the creation of personal folders for storing and categorizing information.

For application developers, the Outlook 2000 application design environment provides advanced design features and tools. Some of these features include the following:

32-bit Forms

You can display and update forms quickly because Outlook 2000 forms are 32-bit and form definitions are usually less than 12 kilobytes.

Instant Switching Between Forms and Run Time

Designers can switch instantly between form design and run time so they can quickly modify and test forms.

Advanced Fields and Views

Fields of an Outlook 2000 form or view can include calculated expressions, validation formulas, and number formatting.

Built-in Forms Modules

Forms Modules support all the features of the built-in Outlook 2000 forms, including digital signatures and encryption. Using a custom form does not mean that you sacrifice standard user interface design and features. You can send custom forms from outside the organization. Users can embed form definitions or layouts in a message so they can send them inside or outside their organization to others who might or might not have the form definition installed.

Extendable Forms

Developers can extend forms with existing ActiveX or OLE controls and with the VBScript language.

The Outlook 2000 design environment enhances the design features offered previously with Microsoft Exchange Client. Outlook 2000 provides tools to create and design folders and to customize forms. These tools include Outlook Forms Designer, Microsoft Visual Basic Scripting Edition (VBScript), and the Microsoft Visual Basic Expression Service.

Outlook Forms Designer

Outlook Forms Designer is the integrated development environment included in Outlook 2000. You can gain access to it within any Outlook 2000 form by choosing **Form** from the **Tools** menu. You can publish forms created using the Outlook Forms Designer in folders and save them as .oft files.

Visual Basic Expression Service

Visual Basic Expression Service is built into the Outlook Forms Designer environment, which allows designers to create validated, formula and combination fields. For example, using Visual Basic Expression Service, the designer can create a formula field for a timecard form that calculates the total number of hours worked in a week.

Instant Collaboration

Instant Collaborations are the built-in application modules included in Outlook 2000. You can customize these modules. To create an instant collaboration you must place a module in an Exchange public folder. The built-in modules are: Calendar, Tasks, Journal, and Contacts. Built-in modules contain predetermined function and design settings, and they require only the services of a public folder to become an instant collaboration.

To create an instant collaboration, you must first create or locate a public folder to contain the module. Then you copy a built-in (Tasks, Contacts, Calendar, Notes or Journal) module into the public folder. After you copy or create a module in a public folder, the combination of the Outlook 2000 module and Exchange public folder results in a collaboration application.

You can create collaboration applications using one of the built-in modules included in Outlook 2000. For example, if you need an application to track individuals or companies, you can use the Contacts module to take advantage of the built-in views and fields.

Built-in applications have a variety of predefined views and you can easily integrate them with the Journal module. Contacts applications offer e-mail and Web integration. For example, users can click on a hyperlink to go to a site on the Internet.

A contacts application provides a group of users with a shared database that stores information about clients, friends, or vendors. To make it a collaboration, users can post it in a public folder.

Creating an application using one of the built-in modules does not require a developer’s expertise. The Outlook Forms Designer and Microsoft Visual Basic Scripting Edition (VBScript) are available for more complex applications.

Calendar Module

When you copy or create the Calendar Module in a public folder users can share, post, and update schedules for activities such as training classes, sporting events and company functions. For example, product launch milestones such as trade shows, product ship dates, or press tours can be posted to a Calendar folder for group viewing.

Tasks Module

When you copy or create the Tasks Module in a public folder team members can share a common task list that displays who is responsible for a task and the status of the task.

For example, a project manager can create a public Tasks folder that team members can update when tasks are completed or significant progress is made. This provides the product manager with up-to-date information on the status of a project.

Contacts Module

When you copy or create the Contacts Module in a public folder users can add to, update, and share a list of contacts. For example, the sales department can share a list of leads, or the entire company can share a list of vendor contacts.

Journal Module

When you copy or create the Journal Module in a public folder users can log and track information such as the amount of time an individual spends on a particular task, on a project, or with a specific customer. You can set the Journal to log and store Office documents, contact calls, email, and other communications.

Notes Module

You can share or privately use the Notes Module, which is the graphical interface equivalent of self-stick notes. As with the other built-in modules, copying or creating this module in a public folder allows users shared access to it. The color and category can be customized for easy retrieval, and the note can be forwarded as a message.

Event Support in Exchange 2000

Events allow applications to define custom actions on particular events. For example, a workflow application can initiate a workflow process when an object is saved (in effect, delivered) to a particular folder. There are three types of events: Protocol events, Transport events and Web Storage System events. Protocol events are actions or occurrences that take place at the protocol level. Transport events extend the message processing system. Web Storage System events are OLE/DB-based Server Events that provide an API through which developers can extend the features of the Web Storage System. Web Storage System events include synchronous, asynchronous and System events. Synchronous events occur before the change is committed to the store. Asynchronous events are essentially notifications and happen after the item is saved or deleted. System events involve only three web storage system-related events: starting the Web Storage System, stopping the Web Storage System, and the timer.

You have control over which users can register event sinks which users can register for events, and what the event sink can access. For more information about Event Security, please go to http://www.Microsoft.com/Exchange.

Using Event Sinks In Exchange 2000

Exchange 2000 enables the user to write code to gain access

to SMTP/NNTP stacks. For example, you might want to add a disclaimer to all out-going e-mail. With Exchange 2000, you can write the same CDO sink in any COM-compatible language. This allows you to create a mail system on top of the Windows 2000 SMTP service, as well as extending Exchange 2000 and Windows 2000 SMTP services.

Developers can build event sinks by creating DLLs either in C++ or Visual Basic. They can also register event sinks using simple script.

Web Storage System Events

Web Storage System events provide a mechanism through which applications can execute code whenever an item is saved, deleted, moved, copied or modified. Web Storage System events run on the server of the client that causes the event to happen. Using Web Storage System events, an application developer can build application logic that executes if the client is a MAPI client such as Outlook, if it is an HTTP client such as Internet Explorer or OWA, or if it is a Win32 client such as any Microsoft Office application. Developers can build application logic that executes if the item is delivered using SMTP, IMAP or NNTP.

Separating the application logic from the client accessing the data and moving it onto the server relieves the developer from the task of rewriting the code for every client that can access the data. It also ensures that the application logic always runs whether the client is connected when the item arrives in the Mailbox Store or not or whether the client has the application installed or not.

The Web Storage System causes a number of different events. These events are either synchronous, asynchronous or system events. Table X.X shows the Web Storage System events that Exchange 2000 supports.

Asynchronous Events	Synchronous Events	System Events
OnSave	OnSyncSave	OnTimer
OnDelete	OnSyncDelete	MDBOnStartUp
		MDBOnShutDown

Table X.X Web Storage System Events that Exchange 2000 Supports

The Web Storage System guarantees that Asynchronous events are delivered. This means that the application sink is called even if the store is shut down before the event fires. When this occurs, the event fires when the store restarts.

Developers can write code in Visual Basic, C++ or any other language that supports COM.

Synchronous Events

An application that processes a synchronous event can modify the item before it saves it to the store, prevents it from being saved to the store or, in the case of a delete event, prevents it from being deleted from the store. Because the application is called before the item is saved, the application has an opportunity to modify the item before any client can access it or is even aware that it is there.

Synchronous events occur within the context of a local transaction. This means that they are called once before, and once after, the transaction is committed. The first time the event sink is called is referred to as the begin phase. During the begin phase the event sink has full read write access to the item that caused the event. The second time

the event sink is called is referred to as the commit or abort phase. During this phase the item is read only.

The Web Storage System provides two synchronous events; synchronous save and synchronous delete. Synchronous save is called whenever an item is saved or delivered into the store. An application that attempts to process synchronous save events must implement an event sink with the following signature:

```
Private Sub IExStoreSyncEvents_OnSyncSave(
ByVal pEventInfo As Exoledb.IExStoreEventInfo,
ByVal bstrURLItem As String,
ByVal lFlags As Long
)
‘ Your code here
End Sub
```

The first parameter contains the event information structure. The second parameter is a string that contains the URL of the Item on which the event is occurring and the third parameter is a long data type containing a series of bits that describe the type of save event and the phase of the event. This provides functions and properties through which to abort the transaction, get an interface with the item being saved, to get an interface with the registration item and to get an interface with the session.

To get an ADO record with the item causing the event, specified in the pEventInfo parameter, you must first get a dispatch interface with the event information structure as follows:

```
Dim DispInfo As IExStoreDispEventInfo
Set DispInfo = pEventInfo
```

Then you simply get the event item as a Record:

```
Dim ado_rec As ADODB.Record
Set ado_rec = DispInfo.EventRecord
```

When you have a Record interface with the event item you can read, write or modify any property of the item. Because you are modifying the item within a synchronous save event, all changes you make to the item occurs before the item is saved to the store.

```
Dim subject As String
subject = ado_rec.Fields("urn:schemas:mailheader:subject")
ado_rec.Fields("urn:schemas:mailheader:subject") = "This is a demo(" &
subject &")"
ado_rec.Fields.Update
```

It is important to understand that new items you save to the Web Storage System do not exist until the transaction is committed. Because the item does not exist, the URL to the item is invalid until it is saved. The following code demonstrates this and shows how to use the flags passed into the event sink to determine what caused the event and the current state of the item being saved.

```
Private Sub IExStoreSyncEvents_OnSyncSave1(
ByVal pEventInfo As Exoledb.IExStoreEventInfo, _
ByVal bstrURLItem As String,
ByVal lFlags As Long)

    If (lFlags And Exoledb.EVT_SYNC_BEGIN) > 0 Then
        If (lFlags And Exoledb.EVT_NEW_ITEM) > 0 Then
            'This is a new item so bstrURLItem is NULL
        Else
            'This is not a new item so
            'bstrURLItem contains the URL to the item
        End If
    Else
        If (lFlags And Exoledb.EVT_SYNC_COMMITTED) > 0 Then
            'The item has been saved
            'so bstrURLItem contains the URL to the item
        Else
            If (lFlags And Exoledb.EVT_SYNC_ABORTED) > 0 Then
                'the item will not be saved
            End If
        End If
    End If


End Sub
```

The first time the event sink is called the flags parameter is set to Exoledb.EVT_SYNC_BEGIN. If the item is saved for the first time the Exoledb.EVT_NEW_ITEM is also set and the URL to the item is NULL. If the item is not a new item the Exoledb.EVT_NEW_ITEM is not set and the URL contains a valid URL to the modified item.

The second time the event sink is called the flags parameter is either set to Exoledb.EVT_SYNC_COMMITTED if the item is committed to the store or Exoledb.EVT_SYNC_ABORTED if the transaction is aborted.

Synchronous delete is called whenever an item is deleted from the store. An application that attempts to process synchronous delete events must implement an event sink with the following signature:

```
Private Sub IExStoreSyncEvents_OnSyncDelete(ByVal pEventInfo As _
Exoledb.IExStoreEventInfo, ByVal bstrURLItem As String, ByVal lFlags As
Long)
```

```
‘ Your code here
End Sub
```

Because an item must exist to be deleted, the URL passed to the event sink always points to a valid item in the store. During the begin phase of a synchronous event sink you can abort the transaction by calling “AbortChange”. To conditionally abort a delete based on a property of the item, you can check the value of the property and then conditionally abort the transaction:

```
Dim DispInfo As IExStoreDispEventInfo
Dim ado_rec As ADODB.Record
Set ado_rec = DispInfo.EventRecord
Set DispInfo = pEventInfo
Dim subject As String
If (InStr( ado_rec.Fields("http://schemas.mycompany.com/myprop"), “ready
to delete”) > 0 Then
DispInfo.AbortChange 1
End If
```

This aborts the transaction and returns the error code passed to the **AbortChange** function. This is particularly useful when controlling the characteristics of a document that is involved in a workflow process. A typical workflow can use ACLs to determine which users can read and write a document and the state of the document. One or more workflow properties on the document (or even on another document), record this to control which users can delete the document.

Asynchronous Events

Asynchronous events happen after the item is saved to the store. You should use them as a notification or when it is not necessary to modify the item before it is saved. Code that executes within a synchronous event must complete before the item can be saved to the store so it is important to ensure that your code does not delay the saving of the item more than is necessary. A typical application defers actions that do not directly affect the item to an asynchronous event. You can change the workflow state of a document within a synchronous event but you should defer updating a web page that advertises the existence and state of the document to an asynchronous event. You should also not typically send out notification mail until the item is saved and the asynchronous event fires.

The Web Storage System provides two asynchronous events; asynchronous save and asynchronous delete. Asynchronous save is called whenever an item is saved or delivered into the store. An application that attempts to process synchronous save events must implement an event sink with the following signature:

```
Private Sub IExStoreAsyncEvents_OnSave(
ByVal pEventInfo As IExStoreEventInfo,
ByVal bstrURLItem As String,
ByVal lFlags As Long)
```

```
‘ Your code here
End Sub
```

The first parameter contains the event information structure. This provides an interface onto the registration. It does not provide an interface onto the event item. To access the item you must bind to it using the URL, passed in the second parameter, using an ADO record:

```
Dim ado_rec As New ADODB.Record
ado_rec.Open bstrURLItem, , adModeRead, adFailIfNotExists
```

Asynchronous delete is called whenever an item is deleted from the store. An application that attempts to process asynchronous delete events must implement an event sink with the following signature:

```
Private Sub IExStoreAsyncEvents_OnDelete(
ByVal pEventInfo As Exoledb.IExStoreEventInfo,
ByVal bstrURLItem As String,
ByVal lFlags As Long)
‘ Your code here
End Sub
```

Registering for Web Storage System Events

When you write and install an event sink on the Web Storage System server you must then register to be called whenever a specific event occurs. To create an event registration you save an item with a content class of “urn:content-class:storeeventreg” into the folder in which you want the event to fire. You achieve this using ADO.

To create a record:

```
Dim rBinding
Set rBinding = CreateObject("ADODB.Record")
rBinding.Open strBindingURL, cSess, 3, 33554432
```

To set the content class to make this a registration item:

```
rBinding.Fields("DAV:contentclass") = "urn:content-class:storeeventreg"
```

To set the name of the event to catch:

```
rBinding.Fields("http://schemas.microsoft.com/exchange/events/
EventMethod") = "OnSyncSave"
```

To set the program ID of the sink that services the event:

```
rBinding.Fields("http://schemas.microsoft.com/exchange/events/SinkClass")
= "MySink.Sink"
```

To enable the event registration and save the registration item to the store:

```
rBinding.Fields("http://schemas.microsoft.com/exchange/events/Enabled") =
bEnabled
rBinding.Fields.Update
rBinding.Close
Set rBinding = Nothing
```

Accessing the Item

When a synchronous event fires, you receive the item that fired the event. You can access this item using ADO or OleDB. This code illustrates how to access a message attachment:

```
Dim cdo_attachment As CDO.IBodyPart
Dim cdo_msg As New CDO.Message
cdo_msg.DataSource.OpenObject ado_rec, cdoAdoRecord
For Each cdo_attachment In cdo_msg.Attachments
Dim attachment_stm As New Stream
```

To create the URL for the attachment:

```
Dim dir As String
Dim newItemURL  As String
Dim filename As String
dir = ado_rec.Fields("DAV:parentname")
filename =
cdo_attachment.Fields("urn:schemas:httpmail:attachmentfilename")
newItemURL = dir & "/" & filename
```

To create a new item in the store:

```
Dim newItem_rec As New ADODB.Record
Dim newItem_stm As New Stream
newItem_rec.Open newItemURL, , adModeReadWrite, adCreateNonCollection
Or
adCreateOverwrite
```

To open a stream onto the new item:

```
newItem_stm.Open newItem_rec, adModeReadWrite, adOpenStreamFromRecord
```

Get a stream onto the attachment

```
Set attachment_stm = cdo_attachment.GetDecodedContentStream
```

To copy the attachment to the new item:

```
attachment_stm.CopyTo newItem_stm
```

To save all:

```
newItem_stm.Flush
attachment_stm.Flush
```

To clean up:

```
Set newItem_rec = Nothing
Set newItem_stm = Nothing
Set attachment_stm = Nothing
Next cdo_attachment
Set cdo_msg = Nothing
```

SMTP Transport Events

SMTP Transport Events extend the message processing system (address resolution, restrictions processing, and message routing). The message is received in its entirety into the SMTP transport and is handed off to the customized code. This code runs on each message, providing read/write access to envelope and body and executes in process with IIS. You can use transport events to pass an incoming or outgoing message through a custom process before the message is stored or relayed to a recipient. Transport events can fundamentally alter the structure of the message such as adding addition information like disclaimers, or pass the message to a compression agent before submitting information to the next destination. Table X.X shows the commonly used transport events in Microsoft Exchange 2000:

Event	Description
OnMessageSubmission	This event fires for every message that is submitted to the system. This occurs whether the message originated from an SMTP client, a MAPI client, or another client. In this event, you can write message-specific events such as virus checking and deleting mail from specific recipients.
OnSyncMessagePreCategorize	This event fires after the submission of a message. The only difference between this event and the OnMessageSubmission event is that no recipients should be added to a message in this event.
OnMessagePostCategorize	This event fires after the categorization of each message. Once this event fires, the categorized properties of a message are available to sinks. For example, if a sink needs to examine each recipient for a message, it should hook onto this event.

Table X.X Commonly Used SMTP Transport Events

SMTP Protocol Events

You use protocol events for billing and charge-back, based on the number of connections and the time of connections. These events are also useful for monitoring the system using Simple Message Transfer Protocol (SMTP) and for implementing new SMTP commands. Protocol Events extend the SMTP protocol and modify existing SMTP protocol commands and responses.

SMTP Transport Components

SMTP services contain the following Transport Components:

- **Store Driver** Provides backing store for MailMsg objects passing through the service, and handles the delivery of local messages.
- **Advanced Queue** Provides the queue management and logic for message delivery, routing, and relay.
- **Categorizer** Provides categorization services for inbound messages, such as distribution list expansion using the LDAP and Active Directory.
- **Router** Provides routing logic for outbound message relay, such as the Next Hop to which the message should be relayed.

For more information about implementing these components, go to http://www.Microsoft.com/Exchange.

Event Bindings

Bindings are stored in the Metabase under /smtpsvc/<instance number>/EventManager. You use ADSI to modify the Metabase to create bindings. These are tools for registration, de-registration, debugging (list binding). The Event dispatcher uses bindings to match sinks to events. The event rules filter the messages to determine which sink is applicable, if any. Then the priority for each event is determined. Event sinks are executed in order of priority. Events with the same level of priority are executed in random order.

SMTP Protocol Event Sinks

You can extend SMTP features by using protocol sinks. For example, you can add a protocol keyword, reject mail from specific recipients, and exchange information with other SMTP event sinks on the other end of an SMTP connection. You can also restrict the size of messages that can be sent in your organization. Each protocol extension is a collection of sinks handling aspects of the intended protocol command.

A sink can implement one or more sink interfaces and fires depending on what category the session is in. Example categories are Inbound (On Inbound Command, On Server Response) and Outbound (Session Start, Message Start, Per-Recipient, Before Data and Session End). The Protocol Event Sinks are as follows:

- ISmtpInCommandSink This is called when inbound an SMTP command is received
- ISmtpOutCommandSink This is called when outbound an SMTP command is sent
- ISmtpServerResponseSink This is called when a server responds to a previously sent command

SMTP and NNTP Transport Event Sinks with CDO

SMTP is an application-level TCP/IP protocol that supports text-oriented e-mail between computers supporting Message Handling Service. NNTP is the standard for Internet exchange of Usenet messages. These services provide customized synchronous transport event sinks. A transport event occurs when message data is transported into or out of the services. All messages go through events, including X.400 and RPC inbound or outbound. One common example of an inbound transport event is the arrival of a message to the SMTP service either over the network or in the SMTP pickup directory. When this transport event occurs, a source for this event notifies transport event sinks, each of which can take some type of action based on the data content of the message. The system performs no other action with the message data after passing it to the sink until the sink completes and returns. While the sink is executing, it has exclusive control of the message data and state.

The architecture for the SMTP and NNTP source and sink event system is based on the Component Object Model (COM). A transport event sink is any COM object that implements the appropriate COM interface for the event. An event source is a process or thread that notifies sinks using the COM run time when a particular event occurs. The source needs no information about the internal details of the sink; its only requirements are that the sink be a COM object and that it exposes the correct COM interface. Table X.X shows a number of transport event sinks.

Event	Interface	Description
OnArrival	**ISMTPOnArrival**	Handles the arrival of messages to the SMTP service
OnPostEarly	**INNTPOnPostEarly**	Handles the initial arrival of newsgroup headers for messages to the NNTP service.
OnPost	**INNTPOnPost**	Handles the arrival of new posts to the NNTP service.
OnPostFinal	**INNTPOnPostFinal**	Handles all new posts that have been committed to disk or stored by the NNTP service

Table X.X Transport Event Sinks

Event Binding Rules

You should limit Protocol Events to certain commands. The rule is that you limit events to the command that you want to hook (either send or respond to):

- To hook MAIL FROM:, hook “MAIL”
- To hook RCPT TO:, hook “RCPT”

Transport Events limit the event call to certain messages. Transport events are case-insensitive and can filter on MAIL FROM=, RCPT TO=, and EHLO=. For example:

```
MAIL FROM= user1@domain; RCPT TO= user2@domain
```

Note These messages cannot be submitted to a web storage system that has a filter. You only use rules for SMTP-messages.

In this example the sink only fires on messages from user1@domain or to user2@domain. It is possible to substitute wildcards in this example (*@domain, user@random*) or have multiple RCPT TO:s, multiple MAIL FROM:s, and multiple EHLOs. In each case, you choose between FROM or TO.

CDO provides transport event scripting hosts that enable you to implement sinks using a scripting language. The process you use is as follows:

1. Write the script code that implements the appropriate event. Just as one sink Component Object Model (COM) class can implement multiple event functions, one script file can contain code for multiple (or all) events.
2. Register the appropriate CDO event scripting host as the event sink. The scripting host in turn dispatches the event function invocation to the scripting engine that runs your script code.
3. In the Sink Binding Properties, set the ScriptName property so that the binding points to the location of the file containing the script you wrote in the first step. The CDO scripting host will execute the appropriate subroutine or function contained in the script when the event occurs.

Fig X.X illustrates the Event Sink Pathway in the Web Storage System.

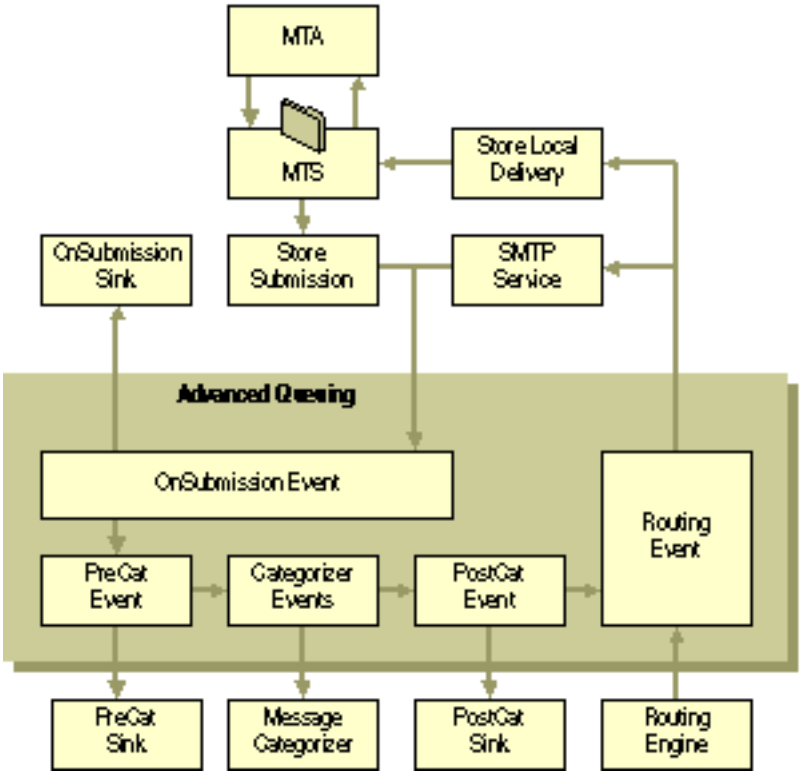

Fig. X.X The Event Sink Pathway in the Web Storage System

Workflow Components For Exchange 2000

Workflow Components provide you with greater control over the impact of application development on your servers. In the past, a developer could accidentally delete files, disrupt operations, and possibly bring the server down with the wrong code. As well, developers are often given unprecedented access to objects on the server, which can raise security concerns. This issue is corrected with the use of Restricted Mode and Privileged Mode in Workflow Designer.

Restricted Mode is granted to developers who are not trusted. In this mode, no compiled objects are permitted. The developer’s code has limited access for binding to, or deleting objects, on the server. This produces a much safer production environment.

Privileged Mode permits developers to perform almost any function on your server, including using compiled objects. Therefore you should only grant this privilege to trusted developers whose applications do not disrupt your server.

Role membership determines who is eligible for a given mode. You grant this membership using the Component Services snap-in to manage Role membership on the Workflow Event Sink COM+ application. For more information about Restricted Mode and Privileged Mode, go to http://www.Microsoft.com/Exchange.

Workflow

The term workflow describes applications that are modeled as business processes. Typical workflow applications include forms routing/approval, document review/publishing, and issue tracking. While you can implement such applications in nearly any programming language or development environment, you can simplify the task with the use of a workflow engine and specialized workflow modeling tools. Such tools allow the overall design, or flow, of a business process to be specified in a simple, high-level representation called a process definition. You can easily modify or extend the process definition without rewriting all of the low-level application code. The workflow engine executes and manages individual instances of a process definition also known as process instances.

Creating Workflow Processes

When you create a workflow process, the workflow designer creates a set of rules for the selected folder and its association with an event sink. The set of rules is called a process definition, the association is called an event registration, and the event sink is the CDO Workflow Event Sink.

Workflow Designer for Exchange simplifies application development. The Workflow engine and event sink are provided with Exchange 2000 server. Workflow applications built for Microsoft Exchange 2000 Server use the object model to create process definitions, consisting of a table of actions, or rules, that define the business process. At runtime, the applications call the workflow engine to create and manage process instances. The workflow engine evaluates the process definition to see which actions should be run, and in turn executes either Microsoft Visual Basic Script code, or compiled COM objects, to perform application-specific business logic.

Using CDO Workflow Event Sinks

Exchange 2000 includes a Workflow Event Sink which is provided to relieve the developer of writing low level code to handle security, the scripting environment, resource sharing and data binding. The CDO Workflow Event Sink is the interface between the Web Storage System process and the workflow engine. It is registered automatically as a COM+ application package when you install Exchange. You register the event sink in a folder, where it intercepts all changes made in that folder, and automatically calls the workflow engine, on your application's behalf, to create and manage process instances. The event sink also hooks into system timer events to automate expiring items, overdue work, or other cleanup tasks. The event sink works well in scenarios where users are creating or editing documents using off-the-shelf applications such as Microsoft Word or Microsoft Excel.

Workflow Designer applications build on the Web Storage System architecture, and must run on a Exchange 2000-based server. You typically install them into an application folder—either a public folder, or private Inbox folder. The association between your folder and the event sink makes it possible for the Web Storage System to notify the event sink when an event occurs in your folder. The event sink in turn calls the workflow engine to handle the work item’s transition.

Authorized folder owners register the Workflow Event Sink in application folders that they own. On a per-server basis, you control which users are authorized to register workflows on that server and which users can write "unlimited" workflow scripts by granting Restricted Mode or Privileged Mode to the developer:

Note Microsoft Exchange Server 5.5 includes Routing Objects libraries—helper objects for building simple document routing applications. While similar to the Workflow Designer in some ways, the Routing Objects are designed only for MAPI-based e-mail routing applications, and don't take advantage of Web Storage System features. Exchange 2000 Server is compatible with existing MAPI applications, including Routing Objects. If you want to build a routing application that runs on both Microsoft Exchange 5.5 and Exchange 2000 Server servers, you should use MAPI and the Routing Objects libraries. If you want to write Web-based workflow applications or use new Web Storage System features, you should use Exchange 2000 Server and Workflow Designer.

The ability to create workflow processes is an essential service provided by the Exchange Workflow Designer. The workflow process is a series of tasks or actions, the order in which they must be performed, permissions defining which users can perform them, and script that is executed for each action. The Exchange Workflow Designer provides a graphical view of the workflow process that you can easily create and edit from within the tool.

In its simplest form, the workflow process automates and enforces the order of the tasks in the process. For example, a user creates a new issue and assigns it to another user, who then resolves the issue and assigns it back to the original user, who then can close the issue. By defining the order of the tasks in the workflow process, the team solution can ensure an issue is resolved before it is closed.

The conceptual model for a workflow process includes states and actions. A *state* is a discrete value of the state property of an item. For example, the state of an issue defines where that issue is in the workflow process, such as Resolved or Closed. An *action* defines the operations that can be performed on an item. One special type of

action is a *transition*, which moves the item from one state to another. Actions are triggered by database changes.

Figure X.X shows a diagram of a workflow process used in the Issue Tracking solution, the boxes represent states and arrows represent actions.

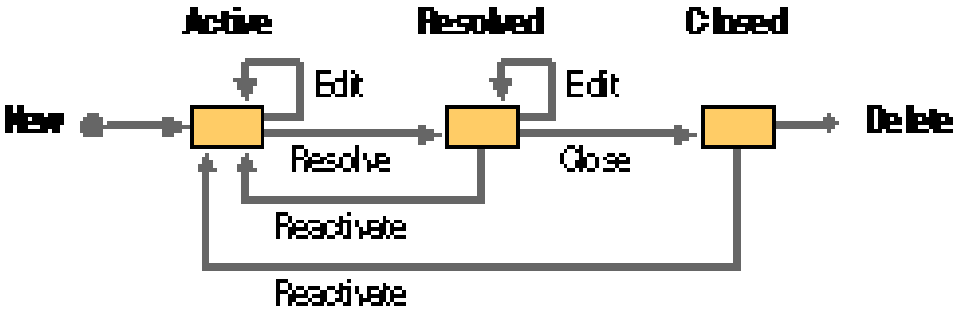

Figure X.X Tracking Solution Workflow Process

Using script with workflow actions can make your workflow process more efficient. Before adding script to workflow, it is important to understand the following:

- Workflow Engine
- Adding the Workflow System Account

Workflow Engine

The workflow engine controls the changes in a document's status in your workflow folder. It runs in response to certain system and Web Storage System events and encapsulates the logic for advancing the state of your workflow documents. This makes it possible for you to control the status of documents you are tracking or guiding.

Exchange 2000 includes an in-process server (CDOWF.DLL) that implements the IProcessInstance Advance function, the workflow engine. The workflow event sink (CDOWFEVT.DLL) calls the engine when an event fires in your workflow-enabled folder. Figure X.X shows the workflow engine process.

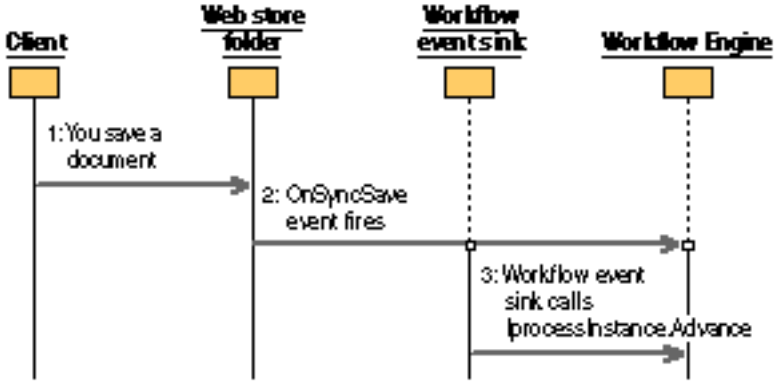

Figure x.x Workflow engine process

You do not call the engine directly unless you are writing an event sink. Other applications can use the workflow engine by making direct calls to the IProcessInstance

Advance function. The event sink passes the engine one of the event types used in your process design as well as useful context information. The engine maintains document integrity based on the parameters the event sink passes.

The engine checks your process design when it receives an event message. You define a workflow process and set it up to be executed based on Exchange Web Storage System events or system events. The process design encapsulates the logic for advancing the state of your workflow documents. It includes script or COM actions and conditions.

The engine attempts to find a match between the message received and a rule in your process logic. When it finds a match it executes the prescribed action. The engine and the script host run in the workflow event sink process and have access to the ProcessInstance and the user context.

When the workflow engine finds a row match and executes the row's action script or COM object, it passes the script host environment a WorkflowSession object with an ADODB Fields collection representing the ProcessInstance (row) undergoing transition. Only the workflow engine can change the status of a ProcessInstance directly. Figure x.x is a diagram that presents the workflow operation concepts:

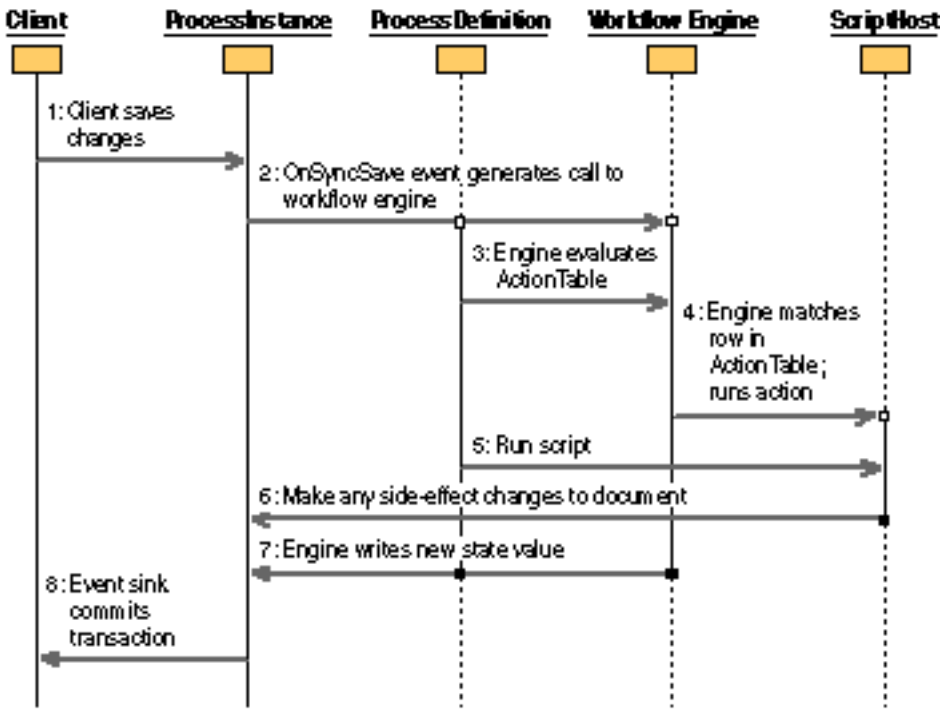

Figure x.x Workflow Operation Concepts

Adding the Workflow System Account

You need to add a Workflow System Account as a member of Domain EXServers group for the workflow event sink to run with proper privileges. You use Active Directory Users and Computers and Component Services to do this. Exchange 2000 Server Setup installs Workflow components by default.

To add the Workflow System Account:

1. Pick an existing account from your Microsoft Windows 2000 domain or create a new account. These instructions assume that you have an account named Workflow System Account.
2. Set the account's password to never expire.
3. Ensure that the Workflow System Account has an Exchange mailbox. If your application uses e-mail-based workflow method (<_cdo_web_based_and_e_mail_workflow_compared.htm> (IWorkflowMessage (<_cdo_iworkflowmessage_interface.htm> SendWorkflowMessage <_cdo_iworkflowmessage_sendworkflowmessage.htm>), then the Workflow System Account mailbox folder must be on the same physical server as the application folder where you are registering the workflows. This restriction does not apply to the IWorkflowMessage Send method (<_cdo_imessage_send.htm>), which uses SMTP.
4. Make the Workflow System Account a member of the Windows 2000 group named Domain EXServers. This enables the CDO Workflow event sink to execute with proper privileges.
5. In the console tree of Component Services, select your computer. Then select the COM+ application named **Workflow Event Sink**. Select Properties from the **Action** menu.
6. Select the **Identity** tab, click **This user**, and type the name and password for the Workflow System Account.
7. Use the Domain Controller Security Policy management console to configure the Workflow Event Sink account to act as part of the operating system. If you are configuring a member server (rather than a domain controller), use the Local Security Policy management console.
8. Add the Workflow System Account to the Privileged Workflow Authors <_cdo_privileged_workflow_authors.htm> role in the COM+ application package for the workflow event sink <_cdo_workflow_event_sink.htm>.

Workflow Security

When an action table includes scripts and COM objects that run on your server, security is an important issue. You need to prevent these scripts and COM objects from causing problems on your server while allowing users the flexibility to design their own workflows. The Workflow system account must have permission to act as part of the operating system, which is administered through the Microsoft Management Console (MMC). Workflow Components solves this problem by implementing workflow security modes. Table x.x summarizes the two modes of security that Workflow Designer implements.

Factors to Consider	Restricted	Privileged
Script Access	Only the current Web Storage System item (ProcessInstance) is	Scripts and objects can access enterprise databases as security

	accessible	context allows.
Security Context	EUSER_EXSTOREEVENT (guest privileges)	Workflow System Account defined by system administrator
CoCreateable COM objects	None	Unlimited registered components Allows you to integrate with other systems such as SQL databases and other business applications that provide COM components. Can use LDAP and Active Directory.

Table x.x Two modes of security that Workflow Designer implements

You use the restricted mode for workflows that are not trusted. A user needs to be able to set up simple document tracking applications. Users should be free to do this, but you don't want them to run any script or COM object.

The security mode is a property of the process definition and you set it at design time. You have the option of either setting this property as restricted or privileged. You must also be a member of PrivilegedWorkflowAuthors group for your workflow to run in privileged mode. This is checked at run-time. PrivilegedWorkflowAuthors is a built-in group, and only you as the network administrator can add a developer to this group.

The workflow engine enforces security by checking privileges at run-time. For privileged mode workflows, it verifies that all the design-time pieces of a workflow are modified last by a member of the PrivilegedWorkflowAuthors role. This includes:

- ProcessDefinition
- CommonScript
- Event Sink Registration Document

The call to Advance (engine) fails if any of these are modified last by any users other than PrivilegedWorkflowAuthors.

Exchange 2000 Server setup creates the COM+ Application package for the Workflow Event Sink and installs the PrivilegedWorkflowAuthors role. If you write your own event sink, you must register it as COM+ Application package in the Component Services tool, and add the PrivilegedWorkflowAuthors role. Your sink's calls to the workflow engine fails when this role is not present in the sink's security context.

Application Development Utilities

Following are some of the Exchange 2000 development utilites that you can use to develop applications

Web Storage System Viewer

The Web Storage System Viewer allows you to see the contents of folders in the Web Storage System.

For more information about this tool, search for Messaging and Collaboration at http://www.Microsoft.com/Exchange.

Web Storage System Schema Designer

The Web Storage System Schema Designer is a tool that allows you and developers to view and modify the schema for folders in the Web Storage System. This tool is written in Visual Basic and you can run it on both the client and the server. It uses WebDav to communicate to the server to retrieve and update the schema. While users of the tool should have an understanding of how schema is supported in Exchange, they don’t need to know the details of how schema information is stored.

For more information about the Web Storage System Schema Designer go to http://www.Microsoft.com/Exchange

The Web Storage System Application Deployment Wizard

The Web Storage System Application Deployment Wizard simplifies deploying applications. The application packaging scans a given virtual root and builds a script that allows you to deploy the application on another server or group of servers. The distribution plays the script and builds the application on another server.

For more information about the Web Storage System Application Deployment Wizard, go to http://www.Microsoft.com/Exchange.

The Event Sink Template Wizard for Visual Basic 6.0

The Event Sink Template Wizard for Visual Basic 6.0 creates a Visual Basic project that implements the necessary components needed to create the DLL. This assists developers in building event sink DLLs in Visual Basic.

For more information about the Event Sink Template Wizard for Visual Basic 6.0, go to http://www.Microsoft.com/Exchange.